



# A Modern Day Language Engineering Framework

Benjamin Friedman Wilson



TypeFox

# A bit about myself...

- Language Engineer @ TypeFox
- Studied CS at Oregon State University (BS + MS)
- Working on supporting Language Engineering stacks



# Overview

- What is **Langium**?
- Goals of **Langium**
- Features of **Langium**
- Running **Langium** in the Web + Demo

# What is Langium?

# What is Langium?

- Lang. engineering framework
- TypeScript + NodeJS

# TypeScript

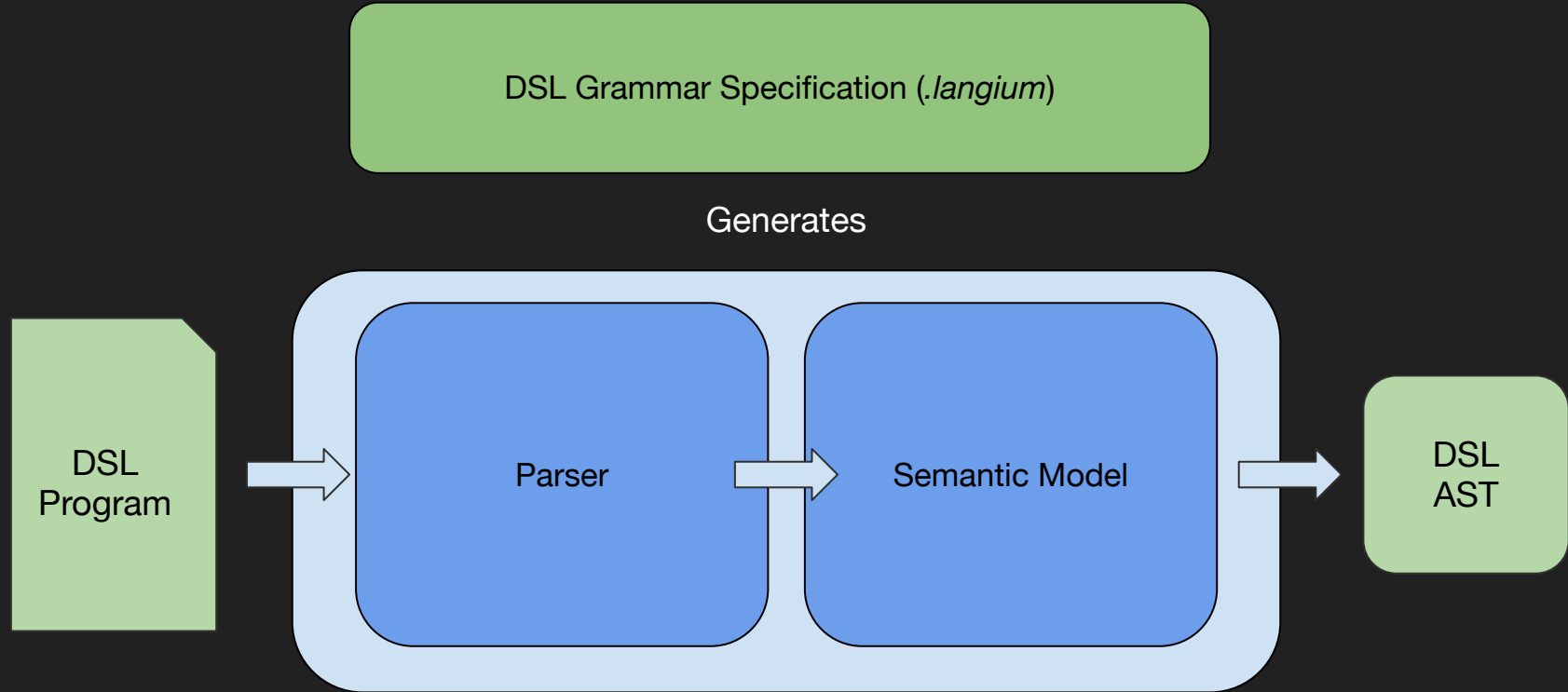


# What is Langium?

- Spiritual successor to Xtext
- Powered by Chevrotain



# What is **Langium**?

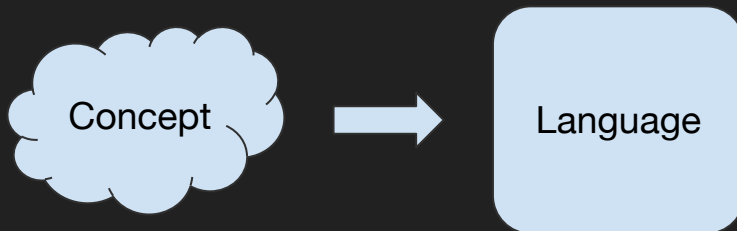


# Goals of Langium



# Goals of Langium

- Low Barrier to Entry
  - Simple Grammar
  - Stays close to TS
  - Quick to get a Language Working



# Goals of Langium

- Lean by Default
  - Include what's *needed*
  - Runs out of the box

Grammar

*... defaults ...*

Validation

Generation

# Goals of Langium

- Customizable by Design

...

*Custom Token Builder*

*Custom Scope Provider*

*More Custom Services...*

# Features of Langium


# Features of Langium

- Cross-References
- Workspace Management
- Language Server Protocol

# Features of Langium

- Cross-References
  - Explicit at *Grammar* level
  - Reference Validation
  - GoTo XYZ Support
  - Symbol Renaming

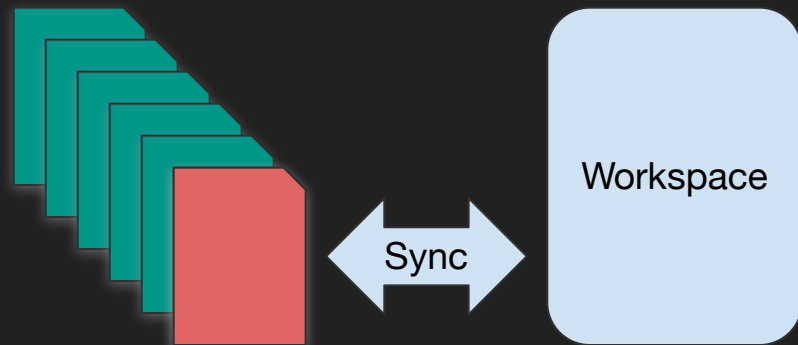
```
Param:  
  name=ID;  
Ref:  
  val=[Param:ID];
```



```
def square(x, y, scale) {  
  move(scale, 0)  
  move(0, scale)  
  move(-1 * scale, 0)  
  move(0, -1 * scale)  
}
```

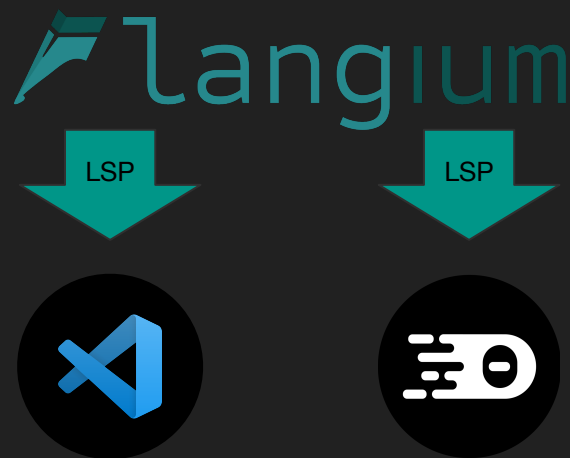
# Features of Langium

- Workspace Management
  - Collection of DSL files
  - Syncs Updates by Heuristic
  - Multi-DSL Workspaces



# Features of Langium

- Language Server Protocol Support
  - VS Code, Theia, NeoVim, BBEdit
  - Diagnostics
  - Code Actions
  - Building VS Code Extensions

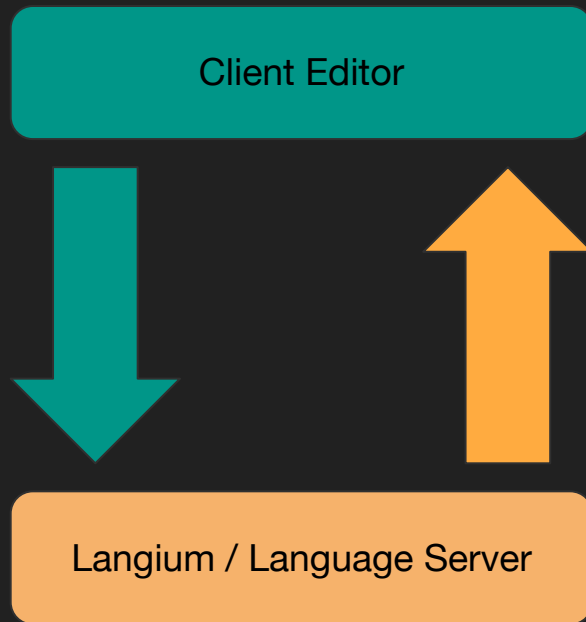




# Running Langium in the Web

# Running **Langium** in the Web

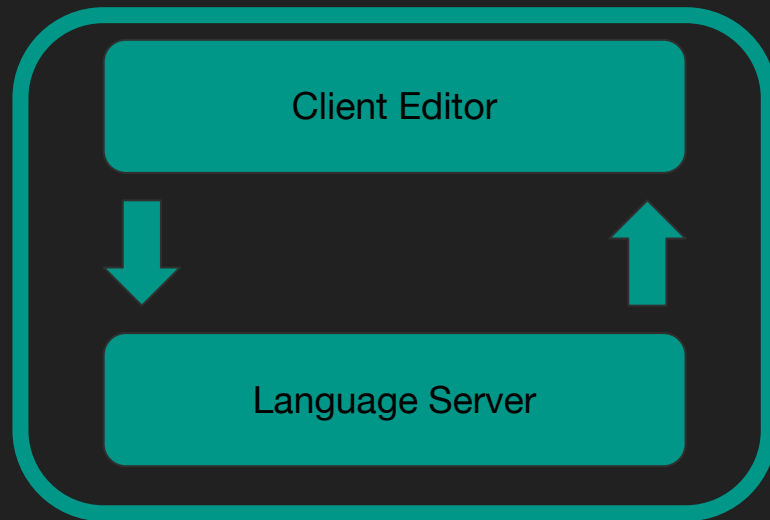
- Server-Client Architecture
- Akin to Desktop Env in 2 Parts
- Limited by Server (connectivity)
- Extra Work to keep it all *together*



But what if these systems  
were the *same*?

# Langium in the Web

- Client-only Architecture
- Editor + Language Server Bundled
- Limited by Client Capabilities
- Limited by App Size



# Langium Demo

# Langium Demo

- Writing *MiniLogo*
- Based off *Logo*

[https://el.media.mit.edu/logo-foundation/  
what-is-logo/logo-programming.html](https://el.media.mit.edu/logo-foundation/what-is-logo/logo-programming.html)

Sketch up a Grammar

Sketch up a Program

*Optional Validation*

*Add Generation*

```

1  /**
2   * test.logo
3
4   * An example MiniLogo program
5   * Draws a simple square on the screen
6   */
7
8   // Draws a square at the given x,y relative offset
9   // Resets back afterwards
10  def square(x, y, scale) {
11
12     move( x, y )
13
14     pen(down)
15
16     move(scale,0)
17     move(0,scale)
18     move(-1 * scale,0)
19     move(0,-1 * scale)
20     pen(up)
21
22     move(-x,-y)
23  }
24
25  square(100,100,100)
26
27  move(100,100)
28
29  color(cyan)
30
31  // draw a 10x10 grid of squares as an example
32  for x = 0 to 10 {
33     for y = 0 to 10 {
34        // call the square routine with a provided x & y start
35        // color(x*10, y*10, (x + y) * 10)
36        square(x * 10, y * 10, 10)
37     }
38  }
39

```

<i>int</i>	::= (any integer)	
<i>var</i>	::= (any variable name)	
<i>macro</i>	::= (any macro name)	
<i>pars</i>	::= $\epsilon$   <i>var</i> , <i>pars</i>	parameter list
<i>args</i>	::= $\epsilon$   <i>expr</i> , <i>args</i>	argument list
<i>cmds</i>	::= $\epsilon$   <i>cmd</i> ; <i>cmds</i>	command sequence
<i>block</i>	::= { <i>cmds</i> }	command block
<i>mode</i>	::= <b>down</b>   <b>up</b>	pen status
<i>expr</i>	::= <i>var</i>	variable reference
	<i>int</i>	literal integer
	<i>expr</i> + <i>expr</i>	addition
	<i>expr</i> * <i>expr</i>	multiplication
	( <i>expr</i> )	grouping
<i>cmd</i>	::= <b>pen mode</b>	change pen mode
	<b>move</b> ( <i>expr</i> , <i>expr</i> )	move pen to a new position
	<b>macro</b> ( <i>args</i> )	call a macro
	<b>for</b> <i>var</i> = <i>expr</i> <b>to</b> <i>expr</i> <i>block</i>	for loop
<i>def</i>	::= <b>macro</b> ( <i>pars</i> ) <i>block</i>	macro definition
<i>prog</i>	::= <b>def</b> * <b>main</b> () <i>block</i>	MiniLogo program

Demo...



After demo...

# Wrapping Up

- Low Barrier to Entry
- Lean & Customizable
- Flexible Deployment



# Langium Going Forward

- [langium.org](https://langium.org)
- Currently 0.4.0 (soon 0.5.0)
- Dev Meetings every Wed. @ 16:00
- [benjamin.wilson@typefox.io](mailto:benjamin.wilson@typefox.io)
- [github.com/montymxb/minilogo-langium-example](https://github.com/montymxb/minilogo-langium-example)



Vielen Dank!

 Langium

  
TypeFox